ADC_Queued_Scan_1 for KIT_AURIX_TC297_TFT

ADC queued source

AURIX™ TC2xx Microcontroller Training
V1.0.1

Please read the Important Notice and Warnings at the end of this document

Scope of work

The Versatile Analog-to-Digital Converter (VADC) is configured to measure multiple analog signals in a sequence using queued request.

The Queued Request of the Versatile Analog-to-Digital Converter (VADC) module is used to continuously scan the analog inputs channels 1, 4 and 5 of group 2.

Introduction

- The Versatile Analog-to-Digital Converter module (VADC) of the AURIX™ TC29x comprises 11 independent analog to digital converters (VADC groups) with up to 8 analog input channels each.

- Each channel can convert analog inputs with a resolution of up to 12-bit.

- Analog/Digital conversions can be requested by several request sources:
  - **Queued request source**, specific to a single group
  - Channel scan request source, which comprises:
    - **Group scan source**, specific to a single group
    - **Background scan source**, which can request all channels of all groups

- A queued source can issue conversion requests for an arbitrary sequence of input channels. The channel numbers for this sequence can be freely programmed.

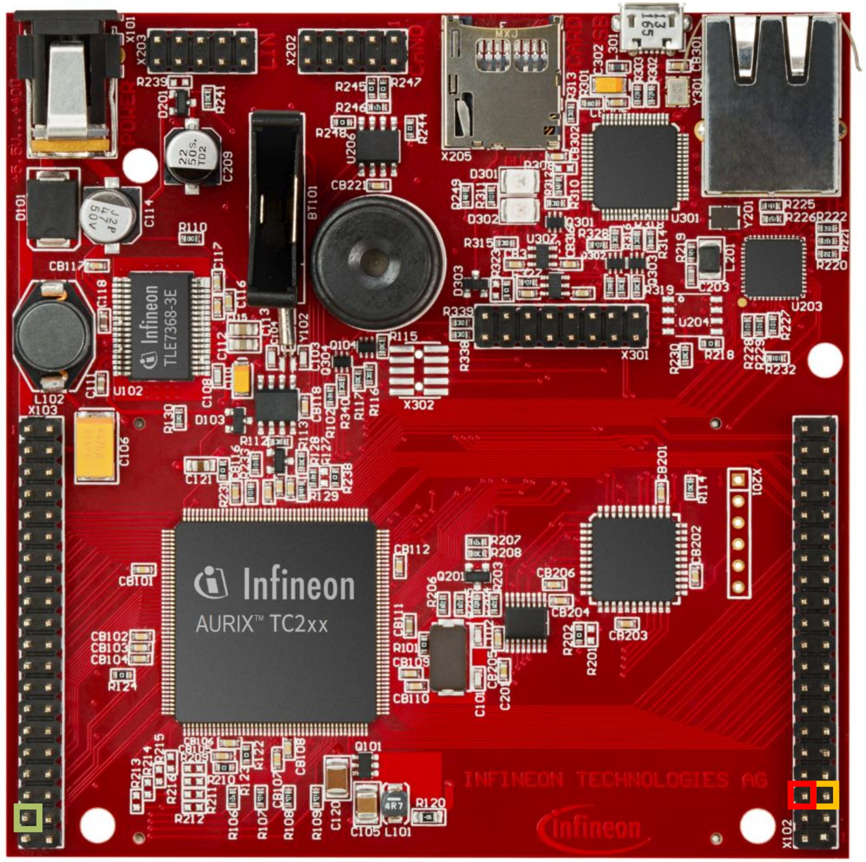
- A queued source converts a series of input channels permanently (using the refill option) or on a regular time base.

Hardware setup

This code example has been developed for the board KIT_AURIX_TC297_TFT_BC-Step.
The signals to be measured have to be connected to channels 1, 4 and 5 of the group 2 of the VADC (pins AN17, AN20, AN21).

X103

VCC_IN	1	2	V_UC(+5V)
GND	3	4	GND
P33.10	5	6	P33.9
P14.8	7	8	P14.7
P14.6	9	10	P10.6
P10.7	11	12	P10.4
P02.0	13	14	P02.1
P02.2	15	16	P02.3
P02.4	17	18	P02.5
P02.6	19	20	P02.7
P02.8	21	22	P00.0
P00.1	23	24	P00.2
P00.3	25	26	P00.4
P00.5	27	28	P00.6
P00.7	29	30	P00.8
P00.9	31	32	P00.10
P00.11	33	34	P00.12
AN45	35	36	AN44
AN17	37	38	AN16
AN25	39	40	AN24



X102

P14.5	40	39	P14.4
P20.10	38	37	P20.9
P15.7	36	35	P15.6
P15.5	34	33	P15.4
P15.3	32	31	P15.2
P22.3	30	29	P22.2
P22.1	28	27	P22.0
P33.11	26	25	P23.4
P23.3	24	23	P23.2
P23.1	22	21	P23.0
P33.6	20	19	P33.8
P33.12	18	17	P33.1
P33.2	16	15	P33.3
P33.4	14	13	P33.5
AN0	12	11	AN8
AN2	10	9	AN3
AN32	8	7	AN33
AN20	6	5	AN21
GND	4	3	GND
V_UC(+5V)	2	1	VCC_IN

Implementation

Configuration of the VADC

The configuration of the VADC is done in the ***initVADC()*** function in four different steps:

- Configuration of the **VADC module**
- Configuration of the **VADC group**
- Configuration of the **VADC channels**
- Filling the queue

Configuration of the VADC module with the function *initVADCModule()*

The default configuration of the VADC module, given by the iLLDs, can be used for this example.
This is done by initializing an instance of the ***IfxVadc_Adc_Config*** structure and applying default values to its fields through the function ***IfxVadc_Adc_initModuleConfig()***.
Then, the configuration can be applied to the VADC module with the function ***IfxVadc_Adc_initModule()***.

Implementation

Configuration of the VADC group with the function *initVADCGroup()*

The configuration of the VADC group is done by initializing an instance of the ***IfxVadc_Adc_GroupConfig*** structure with default values through the function ***IfxVadc_Adc_initGroupConfig()*** and modifying the following fields:

- ***groupId*** – to select which converters to configure
- ***master*** – to indicate which converter is the master. In this example, only one converter is used, therefore it is also the master
- ***arbiter*** – a structure that represents the enabled request sources, which can be Group scan, Queue and/or Background sources. In this example, it is set to ***arbiter.requestSlotQueueEnabled***.
- ***triggerConfig*** – a parameter that specify the trigger configuration

Then, the user configuration is applied through the function ***IfxVadc_Adc_initGroup()***.

Implementation

Configuration of the VADC channels with the function *initVADCChannels()*

The configuration of each channel is done by initializing a separate instance of the ***IfxVadc_Adc_ChannelConfig*** structure with default values through the function ***IfxVadc_Adc_initChannelConfig()*** and modifying the following fields:

- ***channelId*** – to select the channel to configure
- ***resultRegister*** – to indicate the register where the A/D conversion value is stored

Then, the configuration is applied to the channel with the function ***IfxVadc_Adc_initChannel()***.

Filling the queue

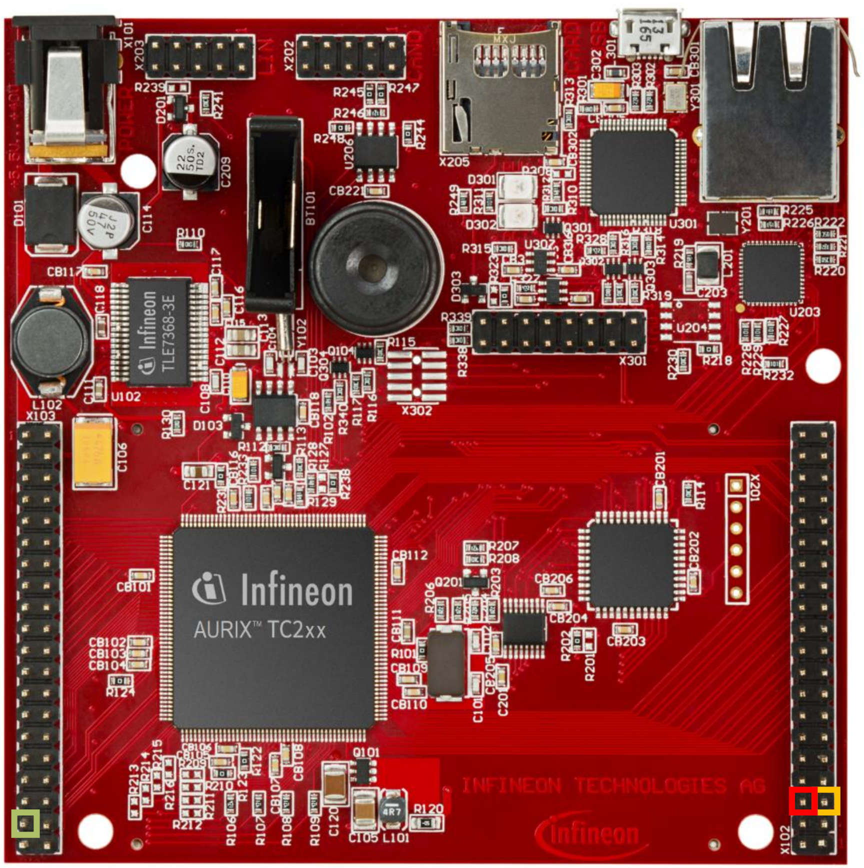
Each channel is added to the queue through the function ***IfxVadc_Adc_addToQueue()***.

When the VADC configuration is done and the queue is filled, the conversion is started with the function ***IfxVadc_Adc_startQueue()***.

Finally, to read a conversion, the function ***IfxVadc_Adc_getResult()*** from iLLDs is used inside the function ***readVADC()***.

All the functions used to get a conversion and configuring the VADC module, its group and channels can be found in the iLLD header ***IfxVadc_Adc.h***.

Run and Test



The signals to be measured have to be connected to channels 1, 4 and 5 of the group 2 of the VADC (pins AN17, AN20, AN21).

Note: For the testing purposes, connect V_UC or GND. Avoid using the VCC_IN (this is the power supplied to the board).

X103

VCC_IN	1	2	V_UC(+5V)
GND	3	4	GND
P33.10	5	6	P33.9
P14.8	7	8	P14.7
P14.6	9	10	P10.6
P10.7	11	12	P10.4
P02.0	13	14	P02.1
P02.2	15	16	P02.3
P02.4	17	18	P02.5
P02.6	19	20	P02.7
P02.8	21	22	P00.0
P00.1	23	24	P00.2
P00.3	25	26	P00.4
P00.5	27	28	P00.6
P00.7	29	30	P00.8
P00.9	31	32	P00.10
P00.11	33	34	P00.12
AN45	35	36	AN44
AN17	37	38	AN16
AN25	39	40	AN24

X102

P14.5	40	39	P14.4
P20.10	38	37	P20.9
P15.7	36	35	P15.6
P15.5	34	33	P15.4
P15.3	32	31	P15.2
P22.3	30	29	P22.2
P22.1	28	27	P22.0
P33.11	26	25	P23.4
P23.3	24	23	P23.2
P23.1	22	21	P23.0
P33.6	20	19	P33.8
P33.12	18	17	P33.1
P33.2	16	15	P33.3
P33.4	14	13	P33.5
AN0	12	11	AN8
AN2	10	9	AN3
AN32	8	7	AN33
AN20	6	5	AN21
GND	4	3	GND
V_UC(+5V)	2	1	VCC_IN

Run and Test

After code compilation and flashing the device, perform the following steps:

- Run the code and then pause it
- Resume step number one to see that the result is changing accordingly to the signal you measure, AN17 is g_results[0], AN20 is g_results[1] and AN21 is g_results[2].

Expression	Type	Value
resultTrace[0].B.RESULT	unsigned short	1908
resultTrace[1].B.RESULT	unsigned short	2451
resultTrace[2].B.RESULT	unsigned short	2583
Add new expression		

References

- AURIX™ Development Studio is available online:
- https://www.infineon.com/aurixdevelopmentstudio
- Use the „*Import...*“ function to get access to more code examples.

- More code examples can be found on the GIT repository:
- https://github.com/Infineon/AURIX_code_examples

- For additional trainings, visit our webpage:
- https://www.infineon.com/aurix-expert-training

- For questions and support, use the AURIX™ Forum:
- https://www.infineonforums.com/forums/13-Aurix-Forum

Revision history

Revision	Description of change
V1.0.1	Update of version to be in line with the code example’s version
V1.0.0	Initial version

Edition 2020-12
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
ADC_Queued_Scan_1_KIT_TC297_TFT